Physically Based Rendering (4th Ed): Chapter 7 Summary

Primitives and Acceleration: The $O(\log N)$ Algorithm

Gemini (For a Developer Audience)

November 29, 2025

Introduction: The Performance Problem

A typical production scene has 10 billion triangles. A typical 4K image has 8 million pixels. If we shoot 1000 samples per pixel, that's 8 billion rays.

Naive Approach: Loop through every ray. Inside that, loop through every triangle.

$$\text{Complexity} = \text{Rays} \times \text{Triangles} \approx 8 \cdot 10^9 \times 10 \cdot 10^9 = 8 \cdot 10^{19} \text{ checks.}$$

This would take years to render a single frame.

Chapter 7 is about **Acceleration Structures**. We need to turn that linear $O(N)$ search into a logarithmic $O(\log N)$ search.

1 Primitives and Aggregates

PBRT uses an elegant object-oriented hierarchy.

1.1 The Primitive Interface

A 'Primitive' is the abstract base class for "things that can be hit." It bridges the gap between pure Geometry (Shapes) and Materials.

- **GeometricPrimitive**: Holds a reference to a Shape (e.g., Triangle) and a Material (e.g., Plastic).
- **Aggregate**: A collection of Primitives that *looks* like a single Primitive.

This is the **Composite Pattern**. You can put a Sphere in a box. You can put that box in a bigger box. The ray doesn't care; it just calls 'Intersect()'.

2 Bounding Boxes (AABB)

The core tool for acceleration is the **Axis-Aligned Bounding Box (AABB)**. It is defined by two points: P_{min} and P_{max}.

The Logic: Checking intersection with a Box is extremely fast.

- If a ray misses the Box, it definitely misses everything *inside* the Box.
- We only check the contents if we hit the Box.

Ray-Box Intersection (Slab Method): A 3D box is just the intersection of 3 pairs of parallel planes (Slabs).

- X-Slab: Region between x_{min} and x_{max}.
- Y-Slab: Region between y_{min} and y_{max}.
- Z-Slab: Region between z_{min} and z_{max}.

The ray hits the box if and only if the intervals of intersection with all three slabs overlap.

3 Bounding Volume Hierarchies (BVH)

The BVH is the industry standard acceleration structure. It is a binary tree.

3.1 Structure

- **Root Node**: A big box containing the entire scene.
- **Internal Nodes**: Have 2 children. They divide the primitives into two groups.
- **Leaf Nodes**: Contain actual triangles (usually 1 to 4).

3.2 Traversal (The Algorithm)

When a ray hits a Node:

1. Does it hit the Node's Bounding Box?
2. **No**: Return false. (Pruned! We just skipped millions of triangles).
3. **Yes**:
    - If Leaf: Check intersection with actual triangles.
    - If Internal: Recursively check Left Child and Right Child.

Optimization: Visit the **closer** child first. If we find a hit at distance $t = 5$ in the close child, and the far child's box starts at distance $t = 10$, we can skip the far child entirely.

4 Building the BVH (SAH)

How do we build a good tree?

- **Bad Split**: Put 1 tiny triangle in Left Child, and 999,999 triangles in Right Child. (Still $O(N)$).
- **Good Split**: Divide primitives so that the cost of traversing both sides is minimized.

PBRT uses the **Surface Area Heuristic (SAH)**. It estimates the "cost" of a split based on probability.

$$Cost = C_{trav} + P(L) \cdot C(L) + P(R) \cdot C(R) \tag{1}$$

- C_{trav}: Cost to traverse a node (memory fetch).
- $P(L)$: Probability that a random ray hits the Left Child box. This is proportional to the **Surface Area** of the box.
- $C(L)$: Cost of the Left Child (number of triangles).

The Algorithm: 1. Sort primitives along an axis (e.g., X-axis). 2. Test different split positions (buckets). 3. Calculate SAH Cost for each split. 4. Pick the split with the lowest cost. 5. Recurse.

Why Surface Area?

Geometric Probability tells us that the probability of a random ray intersecting a convex shape inside a larger box is proportional to its **Surface Area**. Minimizing the surface area of the child boxes minimizes the chance that we have to process them.

Summary for the Developer

1. **The Goal**: Reduce complexity from Linear $O(N)$ to Logarithmic $O(\log N)$.
2. **AABB**: The fundamental unit. Fast to check. If you miss the box, you skip the contents.
3. **BVH**: A tree of boxes.
4. **SAH (Surface Area Heuristic)**: The math used to decide how to split the tree. It balances "balanced tree depth" vs. "minimizing empty space."
5. **Traversal**: Depth-first search. Visit closer nodes first to find a hit early and prune the rest.